REMARKS

Claims 25, 27, 29, 33, 37, and 41 - 43 have been amended. No new matter is introduced with these amendments, which are supported in the specification as originally filed. Claims 25, 27, and 29 - 45 remain in the application.

I. Rejection Under 35 U.S.C. §103(a)

Paragraph 7 of the Office Action dated April 20, 2004 (hereinafter, "the Office Action") states that Claims 25, 27, and 29 - 45 have been rejected under 35 U.S.C. §102(3) as being anticipated by Gorelik et al. (U. S. Patent Publication US2002/0004799) in view of "Serialization of AVL-Binary Tree Element Retrieval via Duplexed Pointers" (March 1992, IBM Technical Disclosure Bullctin, hereinafter "the TDB article").

Applicants have amended all independent claims herein to more specify limitations of their invention. Independent Claims 25 and 27 are amended to specify that two indexes are maintained "to a single data structure". Independent Claims 29, 33, 37, and 41 are amended to specify that "both" data structures (for Claims 29, 33, and 37) or indexes (for Claim 41) represent an initial state. Independent Claim 43 is amended to specify "a single copy of data".

Gorelik, by contrast, uses two separate copies of data (and, as admitted on Page 6 of the Office Action, does not teach using indexes thereto). Applicants respectfully submit that their independent claims are patentably distinct from Gorelik, whether taken singly or in combination with the TDB article. Applicants also respectfully submit that their dependent claims are

patentable by virtue of the allowability of the independent claims from which they depend.

II. Conclusion

Applicants respectfully request reconsideration of the pending rejected claims, withdrawal of all presently outstanding rejections, and allowance of all remaining claims at an early date. (As requested by the Examiner, a clean copy of the claims as currently presented is provided herewith in "Appendix A: Claims as Currently Presented".)

Respectfully submitted,

Marcia L. Doubet

Marcia L. Doubet
Attorney for Applicants
Reg. No. 40,999

Customer Number for Correspondence: 25260
Phone: 407-343-7586
Fax: 407-343-7587

Appendix A: Claims as Currently Presented

Claims 1 - 24 have been canceled.

Claim 25: A method of searching and updating indexes to a data structure in a multi-processing environment, comprising steps of:

maintaining two indexes to a single data structure, a first index for searching and a second index for updating;

responsive to each update of the second index, switching the indexes so that the first index becomes the second index and the updated second index becomes the first index;

allowing searches that are in progress using the first index, before the switching, to continue until completion after the switching, using the newly-switched second index;

after the switching, initiating new searches using the newly-switched first index;

when all searches in the newly-switched second index have completed, updating the newly-switched second index in an identical manner as the update to which the switching step was responsive; and

preventing another operation of the switching step until completion of the step of updating the second index in the identical manner.

Claim 26 has been canceled.

Claim 27: A program product storage medium containing computer instructions that when executed in a computer perform a method of searching and updating indexes to a data structure in

a multi-processing environment, the method comprising steps of:

maintaining two indexes to a single data structure, a first index for searching and a second index for updating;

responsive to each update of the second index, switching the indexes so that the first index becomes the second index and the updated second index becomes the first index;

allowing searches that are in progress using the first index, before the switching, to continue until completion after the switching, using the newly-switched second index;

after the switching, initiating new searches using the newly-switched first index;

when all searches in the newly-switched second index have completed, updating the newly-switched second index in an identical manner as the update to which the switching step was responsive; and

preventing another operation of the switching step until completion of the step of updating the second index in the identical manner.

Claim 28 has been canceled.

Claim 29: A computer program product for serializing data structure retrievals and updates in a multi-processing computer system, the computer program product embodied on one or more computer-readable media and comprising:

computer-readable program code means for creating two identical data structures, both representing an initial state for accessing a single copy of stored data;

computer-readable program code means for performing searches against a first of the two

data structures, the computer-readable program code means for performing searches further comprising a first program instruction for incrementing a search use count for the first data structure atomically during each search to ensure no interference from other processes during that search and a second instruction for decrementing the search use count for the first data structure atomically after performing each search;

computer-readable program code means for performing a first update against a second of the two data structures, yielding a revised data structure;

computer-readable program code means for switching the first data structure and the revised data structure, responsive to completion of the computer-readable program code means for performing the first update, such that the first data structure becomes the second data structure and the revised data structure becomes the first data structure, the computer-readable program code means for switching the data structures further comprising a third instruction for re-ordering data structure pointers atomically to prevent interference from other processes during operation of the computer-readable program code means for switching; and

computer-readable program code means for applying, after operation of the computer-readable program code means for switching, the first update against the second data structure, yielding a second data structure that is structurally identical to the first data structure;

the computer-readable program code means for performing searches further comprising computer-readable program code means for activating the computer-readable program code means for applying the first update against the second data structure when the search use count for the second data structure has a value indicating that no searches are being performed against the second data structure.

Claim 30: The computer program product according to Claim 29, further comprising:

computer-readable program code means for obtaining an exclusive lock on the second data structure prior to operation of the computer-readable program code means for performing the first update; and

computer-readable program code means for releasing the exclusive lock after operation of the computer-readable program code means for applying the first update.

Claim 31: The computer program product according to Claim 29, wherein the computer-readable program code means for performing the first update further comprises computer-readable program code means for queuing a transaction that specifies one or more data structure traversals and one or more data structure modifications that were performed to yield the revised data structure, and wherein the computer-readable program code means for applying the first update further comprises computer-readable program code means for performing the one or more data structure traversals and the one or more modifications specified in the queued transaction against the second data structure that results from operation of the computer-readable program code means for switching.

Claim 32: The computer program product according to Claim 29, further comprising computer-readable program code means for performing a subsequent update against the second data structure that results from operation of the computer-readable program code means for applying the first update; and wherein operation of the computer-readable program code means for

performing the subsequent update causes another operation of the computer-readable program code means for switching and the computer-readable program code means for applying.

Claim 33: A computer system for serializing data structure retrievals and updates in a multi-processing computer system, the computer system comprising:

means for creating two identical data structures, both representing an initial state for accessing a single copy of stored data;

means for performing searches against a first of the two data structures, the means for performing searches further comprising means for incrementing a search use count for the first data structure atomically during each search to ensure no interference from other processes during that search and means for atomically decrementing the search use count for the first data structure after performing each search;

means for performing a first update against a second of the two data structures, yielding a revised data structure;

means for switching the first data structure and the revised data structure, responsive to completion of the means for performing the first update, such that the first data structure becomes the second data structure and the revised data structure becomes the first data structure, the means for switching the data structures further comprising means for re-ordering data structure pointers atomically to prevent interference from other processes during operation of the means for switching; and

means for applying, after switching the data structures, the first update against the second data structure, yielding a second data structure that is structurally identical to the first data

structure;

the means for performing searches further comprising means for activating the means for applying the first update against the second data structure when the search use count for the second data structure has a value indicating that no searches are being performed against the second data structure.

Claim 34: The system according to Claim 33, further comprising:

means for obtaining an exclusive lock on the second data structure prior to operation of the means for performing the first update; and

means for releasing the exclusive lock after operation of the means for applying the first update.

Claim 35: The system according to Claim 33, wherein the means for performing the first update further comprises means for queuing a transaction that specifies one or more data structure traversals and one or more data structure modifications that were performed to yield the revised data structure, and wherein the means for applying the first update further comprises means for performing the one or more data structure traversals and the one or more data structure modifications specified in the queued transaction against the second data structure that results from operation of the means for switching.

Claim 36: The system according to Claim 33, further comprising means for performing a subsequent update against the second data structure that results from operation of the means for

applying the first update; and wherein operation of the means for performing the subsequent update causes another operation of the means for switching and the means for applying.

Claim 37: A method for serializing data structure retrievals and updates in a multi-processing computer system, comprising steps of:

creating two identical data structures, both representing an initial state for accessing a single copy of stored data;

performing searches against a first of the two data structures, the performing searches step further comprising the step of incrementing a search use count for the first data structure atomically during each search to ensure no interference from other processes during the search and the step of decrementing the search use count for the first data structure atomically after performing each search;

performing a first update against a second of the two data structures, yielding a revised data structure;

switching the first data structure and the revised data structure, responsive to completion of the step of performing the first update, such that the first data structure becomes the second data structure and the revised data structure becomes the first data structure, the step of switching the data structures further comprising the step of re-ordering data structure pointers atomically to prevent interference from other processes during operation of the switching step; and

applying, after the switching step, the first update against the second data structure, yielding a second data structure that is structurally identical to the first data structure;

the step of performing searches further comprising the step of activating the step of applying the first update against the second data structure when the search use count for the second data structure has a value indicating that no searches are being performed against the second data structure.

Claim 38: The method according to Claim 37, further comprising steps of:

obtaining an exclusive lock on the second data structure prior to performing the first update; and

releasing the exclusive lock after applying the first update.

Claim 39: The method according to Claim 37, wherein the step of performing the first update further comprises the step of queuing a transaction that specifies one or more data structure traversals and one or more data structure modifications that were performed to yield the revised data structure, and wherein the step of applying the first update further comprises the step of performing the one or more data structure traversals and the one or more data structure modifications specified in the queued transaction against the second data structure that results from operation of the switching step.

Claim 40: The method according to Claim 37, further comprising the step of performing a subsequent update against the second data structure that results from applying the first update; and wherein the step of performing the subsequent update causes repeating the switching step and the applying step.

Claim 41: A method for serializing data retrievals and updates in a computing environment, comprising steps of:

creating two identical indexes, both representing an initial state for accessing stored data and each indexing a single copy of the stored data;

performing searches against a first of the two indexes;

performing a first update against a second of the two indexes, yielding a revised index;

serializing information describing a traversal path taken through the second index for making the first update and one or more modifications made to the second index in the first update;

switching the first index and the revised index, responsive to performing the first update, such that the first index becomes the second index and the revised index becomes the first index;

applying, after the switching step, the first update to the second index, using the serialized information describing the traversal path and the one or more modifications to traverse and modify the newly-switched second index, thereby yielding a second index that is synchronized with, and structurally identical to, the first index; and

performing subsequent searches against the first index.

Claim 42: The method according to Claim 41, further comprising the step of performing a subsequent update against the second index that results from applying the first update; and wherein the step of performing the subsequent update causes repeating the serializing, switching, and applying steps.

Claim 43: A method of serializing access to data in a computing system, comprising steps of:

maintaining two trees as indexes to a single copy of data, a first of which is used for searches and a second of which is used for update operations, each tree having a use count associated therewith;

carrying out searches using the search tree, further comprising the steps of:

determining, for each new search request, which of the trees is currently the search tree;

incrementing the use count for the search tree;

performing the new search request using the search tree; and

decrementing the use count for the search tree, responsive to completion of the performing step; and

carrying out each update using the update tree, further comprising the steps of:

determining which of the trees is currently the update tree;

performing an update to the update tree;

serializing a record describing the update to the update tree;

switching the update tree to become the search tree and the search tree to become the update tree, responsive to completion of the steps of performing the update and serializing the record; and

applying the serialized record to the newly-switched update tree, provided that the use count for the newly-switched update tree has reached a value that indicates that no search requests are currently being performed against this newly-switched update tree, delaying the step

of applying the serialized record if necessary until the use count for the newly-switched update tree has reached this value, and wherein the step of applying the serialized record ensures that both the search tree and the update tree reflect each update.

Claim 44: The method according to Claim 41, wherein the indexes are implemented as trees.

Claim 45: The method according to Claim 41, wherein the indexes are implemented as hash tables.